



WEB PERFORMANCE

오주의 마법사

김민준 류성탁 이솔잎



메인 페이지 왜 이렇게 느려?

메인 페이지가 로딩되는게 눈에 보여!
이거 좀 버벅거리는 것 같은데요

그래도..메인 페이지인데 좀 더 빠르게 할 수 없을까?

Reservation System Performance



Lighthouse

Lighthouse 실행 환경

Runtime environment

- User agent: **Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.101 Safari/537.36**
- Device Emulation Nexus 5X: **Enabled**
- Network Throttling 562.5ms RTT, 1.4Mbps down, 0.7Mbps up: **Enabled**
- CPU Throttling 4x slowdown: **Enabled**

모바일 환경

네트워크 562.5ms RTT / 다운로드 속도 1.4Mb / 업로드 속도 0.7Mb

CPU 4배 성능 저하상태

Reservation System

▶ First meaningful paint 5,610 ms

▶ First Interactive (beta) 6,410 ms

▶ Consistently Interactive (beta) 21,320 ms

▶ Perceptual Speed Index: 15,928 (target: < 1,250)

6

▶ Estimated Input Latency: 30 ms (target: < 50 ms)

100

Opportunities

These are opportunities to speed up your application by optimizing the following resources.

▶ Serve images as WebP

6,990 ms
1,283 KB

Offscreen images

5,510 ms
1,011 KB

Optimize images

4,210 ms
773 KB

Properly size images

2,410 ms
442 KB

Reduce render-blocking stylesheets

2,150 ms

Reduce render-blocking scripts

1,950 ms

Enable text compression

1,810 ms
332 KB

문제1. Offscreen Images

문제2. Optimize Images

문제3. Properly size Images

문제4. Text Compression

왜 이것들이 문제가 될까?

불필요하게 리소스 크기가 커지면

다운로드 시간이 길어지고

결과적으로 렌더링 시간에 영향을 미친다.

첫번째 개선

문제1. Offscreen Images

문제2. Optimize Images

문제3. Properly size Images

문제4. Text Compression

Offscreen Image 란?

화면에 보이지 않는 이미지

페이지가 로드 되는 시점에서 화면에 보이지 않는 이미지들로
인해서 사용자 입장에서 **불필요한 다운로드**가 발생된다.

IntersectionObserver

– Observer에 등록된 DOM element들이 화면에 나타나거나 사라질 때 비동기로 이벤트를 발생시킨다.

– 브라우저 지원 현황

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
		1 52	58			9.3		4.4	
	14	1 54	59		46	10.2		4.4.4	
11	15	55	60	10.1	47	10.3	all	56	59
	16	56	61	11	48	11			
		57	62	TP	49				
		58	63						

```
function _settingIntersectionObserver() {
    var interectionObserver = new IntersectionObserver
    (function (entries, observer) {
        entries.forEach(function (entry) {
            if (!entry.isIntersecting) {
                return;
            }

            var target = entry.target;
            var $lazyImg = $(target).find('img');

            if ( $lazyImg.data('lazy-img') ){
                var source = $lazyImg.data('lazy-img');
                $lazyImg.attr('src', source);
                $lazyImg.removeAttr('data-lazy-img');
                observer.unobserve(target);
            }

        });
    });

    Array.from($productContainer.find('li.item'))
    .forEach(function (el) {
        interectionObserver.observe(el);
    });
}
```

1) IntersectionObserver 설정

3) DOM이 노출되면 이미지 로딩

4) 보여진 이미지는 Observer에서 제거

2) 화면에 나타나는 것을 감지하기 위한
Element를 등록

두번째 개선

~~문제1. Offscreen Images~~

문제2. Optimize Images

문제3. Properly size Images

문제4. Text Compression

이미지는 웹페이지에서 다운로드 되는 바이트의 대부분을 차지한다.

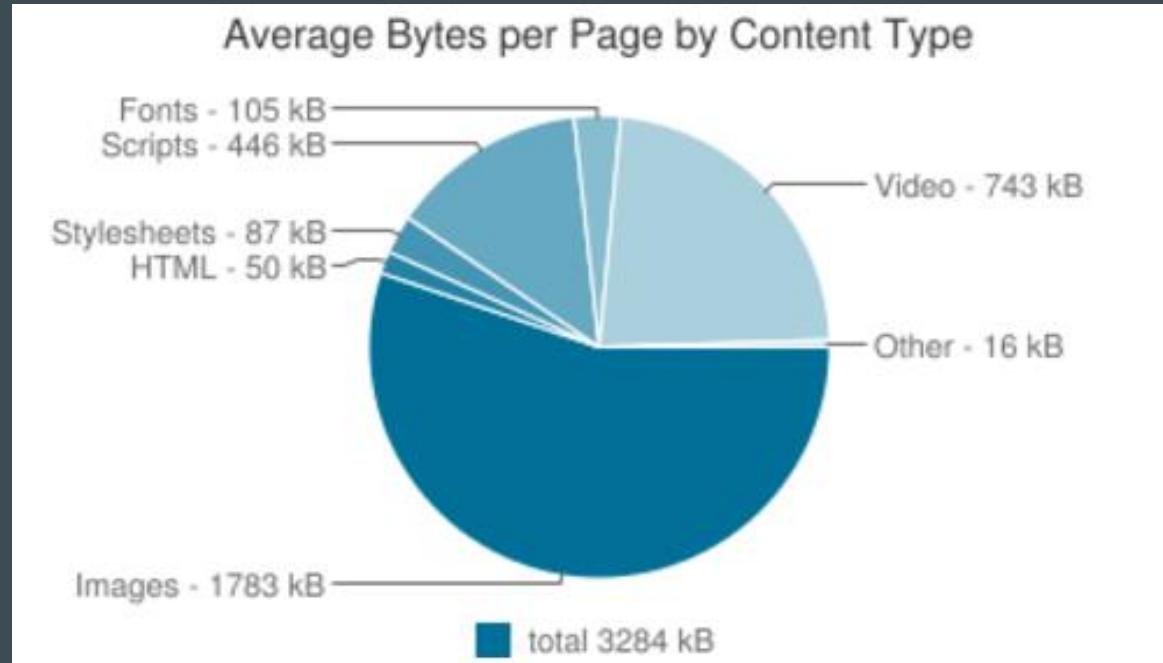


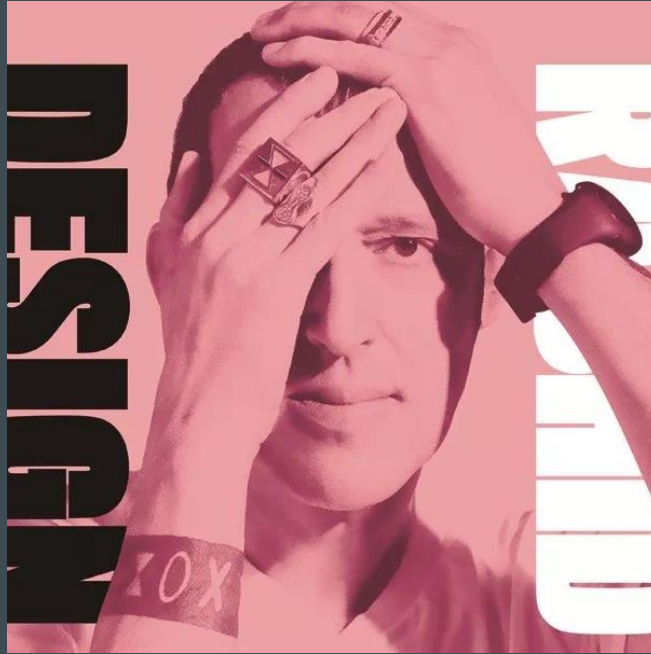
Image File = 총 픽셀 수 X 바이트 수

Image Compression – 각 이미지 픽셀을 인코딩하는 데 사용되는 바이트의 수를 최적화

Image resize – 총 픽셀 수를 최적화

Image Compression

80% 퀄리티 jpg 압축



Before

▶ Optimize images

4,210 ms

773 KB

▶ Properly size images

2,410 ms

442 KB

퀄리티 80%
Jpg 압축

3 Passed Audits

▶ Optimize images

PASS!

Image Resize

실제 이미지 크기 1500 X 1500

불필요한 픽셀 낭비!



예매자 한줄평

★★★★☆ 4 / 5.0

오버헤드 발생!

4.0 | sollip | 2017.08.26. 방문

필요한 ^{4등} 이미지
90 x 90



이미지 파일 압축 (ImageIO)

```
File input = new File(saveFileName);
BufferedImage image = ImageIO.read(input);

File compressedImageFile = new File(saveFileName+".min");
FileImageOutputStream os = new FileImageOutputStream(compressedImageFile);

Iterator<ImageWriter> writers = ImageIO.getImageWritersByFormatName("jpg");
ImageWriter writer = (ImageWriter) writers.next();

writer.setOutput(os);

ImageWriteParam param = writer.getDefaultWriteParam();




param.setCompressionMode(ImageWriteParam.MODE_EXPLICIT);
param.setCompressionQuality(0.8f);
writer.write(null, new IIOMemorySource(image, null, null), param);

os.close();
writer.dispose();
f.setFileName("min_"+originalFilename);
f.setSaveFileName(saveFileName + ".min");
f.setType(3);
fileList.put(fileDao.insert(f), 3);
```

썸네일 (Thumbnails)

```
Thumbnails.of(new File(saveFileName))
    .size(90, 90)
    .outputFormat("jpg")
    .toFiles(Rename.SUFFIX_DOT_THUMBNAIL);
```

ImageIO와 Thumbnails 결과

이름	수정한 날짜	유형	크기
 f0e57ecd-e96b-4fe8-8c5e-21a307a4d67e	2017-08-26 오후 6...	파일	원본 이미지 428KB
 f0e57ecd-e96b-4fe8-8c5e-21a307a4d67e.min	2017-08-26 오후 6...	MIN 파일	압축 이미지 307KB
 f0e57ecd-e96b-4fe8-8c5e-21a307a4d67e.thumbnail	2017-08-26 오후 6...	JPG 파일	썸네일 이미지 3KB

Before

= "short_review_area">



이미지 크기

1500 x 1500 , 428KB

픽셀 수

2,250,00픽셀

(불필요한 픽셀 2,247,200)

After



이미지 크기

90 x 90, 3KB

픽셀 수

1800픽셀

2,247,200 픽셀 감소, 바이트 크기 약 425KB감소

세번째 개선

~~문제1. Offscreen Images~~

~~문제2. Optimize Images~~

~~문제3. Properly size Images~~

문제4. Text Compression

왜 text compression을 해야 할까?

개발 시 필요한 **주석, 공백, 긴 변수명 !**

클라이언트에겐 **불필요한 리소스!**



webpack

style.css	200	stylesheet	(index)	88.1 KB	12 ms
require.js	200	script	(index)	84.7 KB	30 ms
spr_bi.png	200	png	(index)	9.3 KB	48 ms
spr_book_event.png	200	png	(index)	1.7 KB	47 ms
spr_book2.png	200	png	(index)	55.4 KB	44 ms
image.jpg	200	jpeg	(index)	176 KB	290 ms
image.jpg	200	jpeg	(index)	277 KB	294 ms
main.js	200	script	require.js:1962	9.2 KB	50 ms
jquery.min.js	200	script	require.js:1962	84.8 KB	94 ms
hbs.js	200	script	require.js:1962	22.1 KB	26 ms
component.min.js	200	script	require.js:1962	3.9 KB	33 ms
util.js	200	script	require.js:1962	978 B	23 ms
handlebars.js	200	script	require.js:1962	152 KB	160 ms
underscore.js	200	script	require.js:1962	36.2 KB	152 ms
json2.js	200	script	require.js:1962	12.0 KB	144 ms
flicking_component.js	200	script	require.js:1962	9.2 KB	56 ms

Before

리소스 크기 : 500KB 이상

네트워크 요청 수 : 11회

시간 : 500ms 이상

127.0.0.1	200	document	Other	5.7 KB	7 ms
style.css	200	stylesheet	(index)	71.7 KB	15 ms
main.bundle.js	200	script	(index)	108 KB	22 ms
image.jpg	200	jpeg	(index)	176 KB	228 ms
spr_bi.png	200	png	(index)	9.3 KB	14 ms
spr_book_event.png	200	png	(index)	1.7 KB	13 ms
spr_book2.png	200	png	(index)	55.4 KB	13 ms
image.jpg	200	jpeg	(index)	277 KB	224 ms

After

리소스 크기 : 180KB

네트워크 요청 수 : 2회

시간 : 37ms 이상

style.css	200	stylesheet	(index)	88.1 KB	12 ms
require.js	200	script	(index)	84.7 KB	30 ms
spr_bi.png	200	png	(index)	9.3 KB	48 ms
spr_book_event.png	200	png	(index)	1.7 KB	47 ms
spr_book2.png	200	png	(index)	55.4 KB	44 ms
image.jpg	200	jpeg	(index)	176 KB	290 ms
image.jpg	200	jpeg	(index)	277 KB	294 ms
main.js	200	script	require.js:1962	9.2 KB	50 ms
jquery.min.js	200	script	require.js:1962	84.8 KB	94 ms
hbs.js	200	script	require.js:1962	22.1 KB	26 ms
component.min.js	200	script	require.js:1962	3.9 KB	33 ms
util.js	200	script	require.js:1962	978 B	23 ms
handlebars.js	200	script	require.js:1962	12 KB	160 ms
underscore.js	200	script	require.js:1962	10 KB	152 ms
json2.js	200	script	require.js:1962	12.0 KB	144 ms
flicking_component.js	200	script	require.js:1962	9.2 KB	56 ms

Before

리소스 크기 : 500KB 이상
시간 : 500ms 이상

리소스 크기 약 64% 감소, 네트워크 요청 수 9회

시간 약 92%절약!

AFTER

127.0.0.1	200	document	Other	5.7 KB	7 ms
style.css	200	stylesheet	(index)	71.7 KB	15 ms
main.bundle.js	200	script	(index)	108 KB	22 ms
image.jpg	200	jpeg	(index)	176 KB	228 ms
spr_bi.png	200	png	(index)	9.3 KB	14 ms
spr_book_event.png	200	png	(index)	1.7 KB	13 ms
spr_book2.png	200	png	(index)	55.4 KB	13 ms
image.jpg	200	jpeg	(index)	277 KB	224 ms

리소스 크기 : 180KB
시간 : 37ms 이상

최종 성능 개선

Performance

These encapsulate your app's performance.

39

Metrics

These metrics encapsulate your app's performance across a number of dimensions.



First meaningful paint 5,610 ms

First Interactive (beta) 6,410 ms

Consistently Interactive (beta) 21,320 ms

Perceptual Speed Index: 15,928 (target: < 1,250)

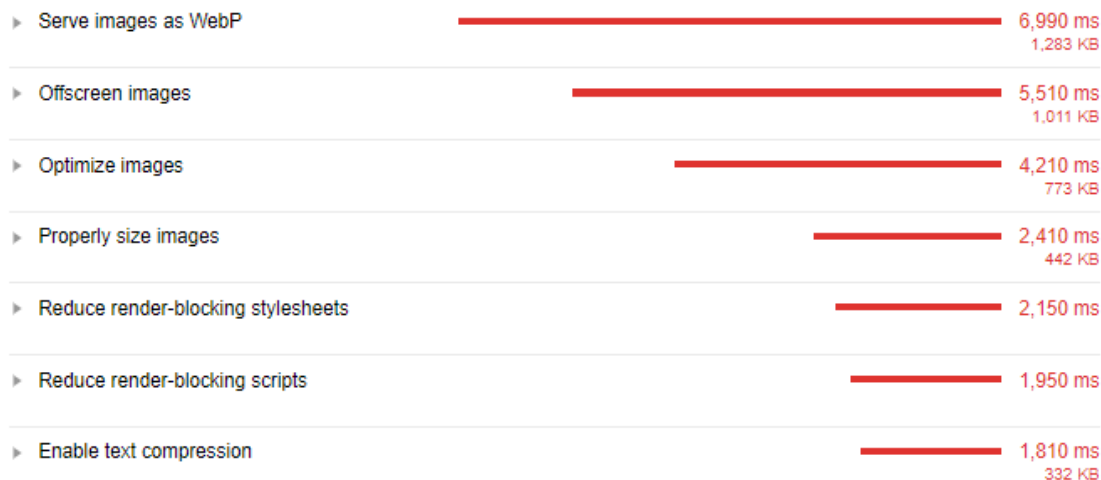
6

Estimated Input Latency: 30 ms (target: < 50 ms)

100

Opportunities

These are opportunities to speed up your application by optimizing the following resources.



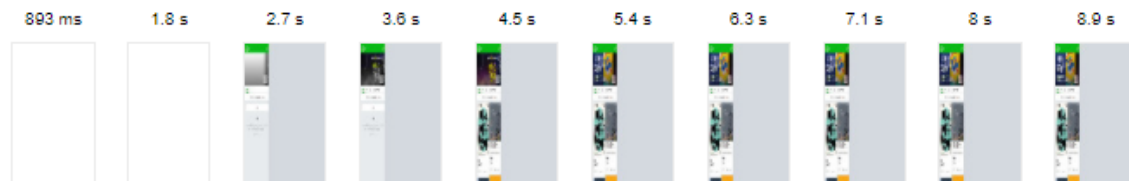
Performance

These encapsulate your app's performance.

92

Metrics

These metrics encapsulate your app's performance across a number of dimensions.



First meaningful paint 2,030 ms

First Interactive (beta) 2,790 ms

Consistently Interactive (beta) 2,790 ms

Perceptual Speed Index: 4,741 (target: < 1,250)

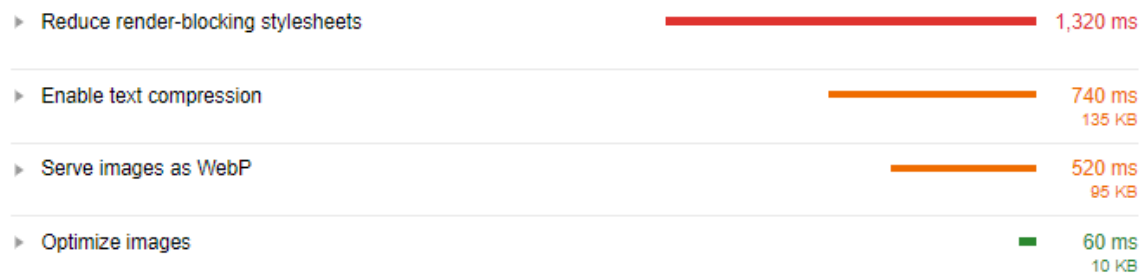
58

Estimated Input Latency: 26 ms (target: < 50 ms)

100

Opportunities

These are opportunities to speed up your application by optimizing the following resources.



6 Passed Audits

- Reduce render-blocking scripts
- Properly size images
- Offscreen images

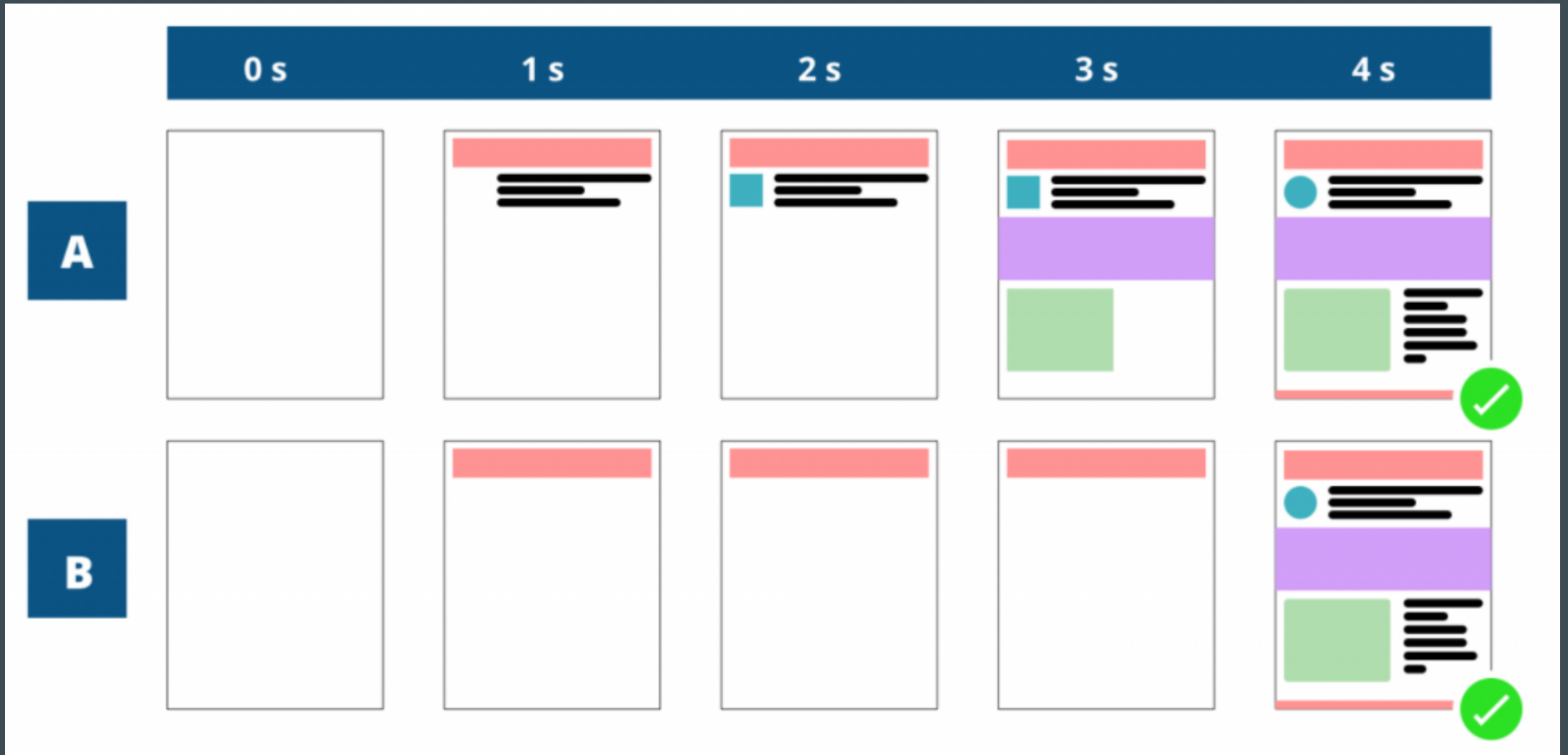
Speed Index

기존의 웹 성능 측정 방식
“브라우저가 **onload**이벤트에 도달하는 시간“

But, 이 수치는 사용자 경험이 반영되지 않는 방법이다.
그래서 나온 새로운 웹 성능 측정 방식 “**Speed Index**”

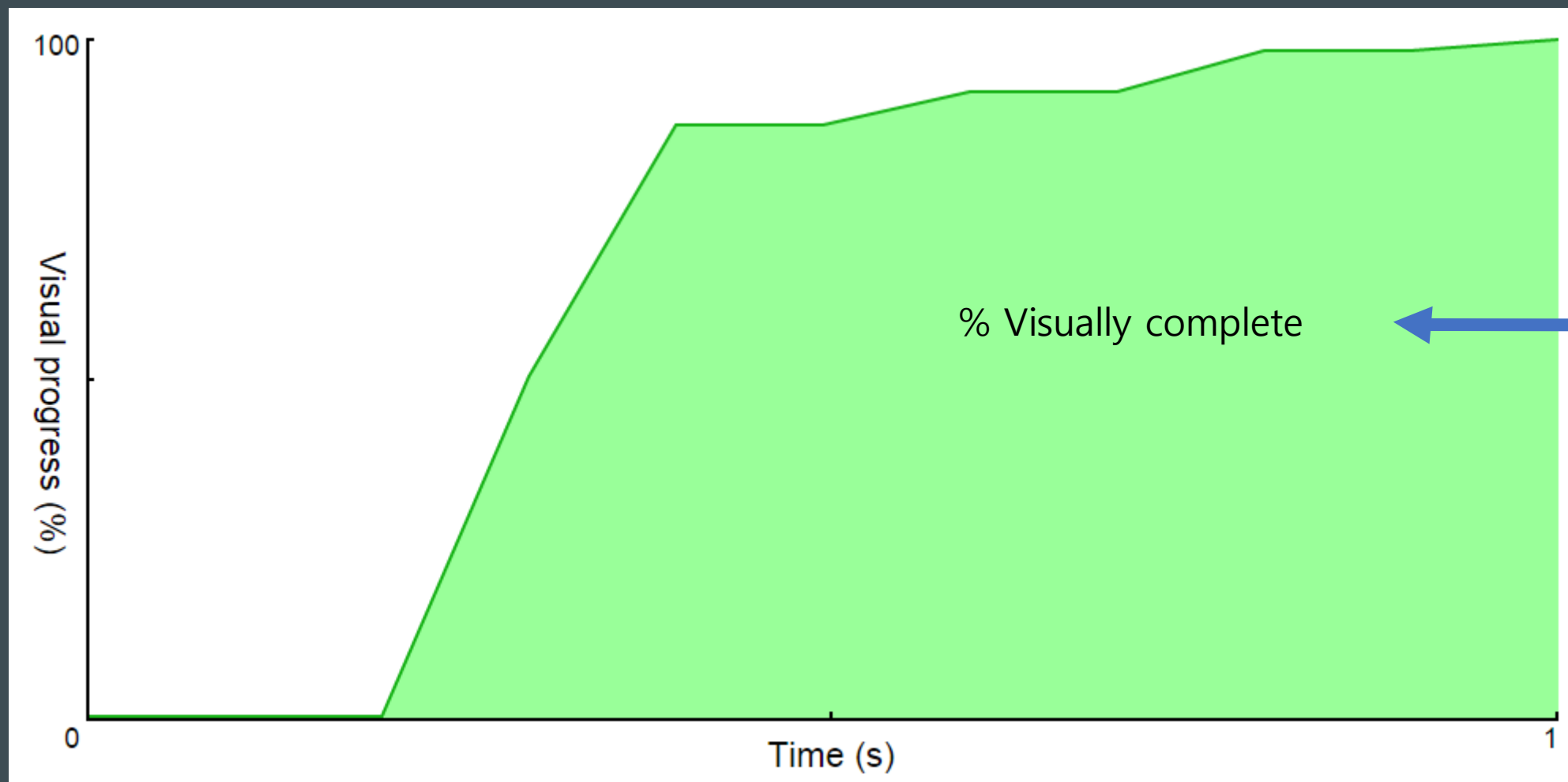
Speed index는
“**실제** **사용자의 경험**을 포함한 평균 페이지 로딩시간”

Speed Index 예시

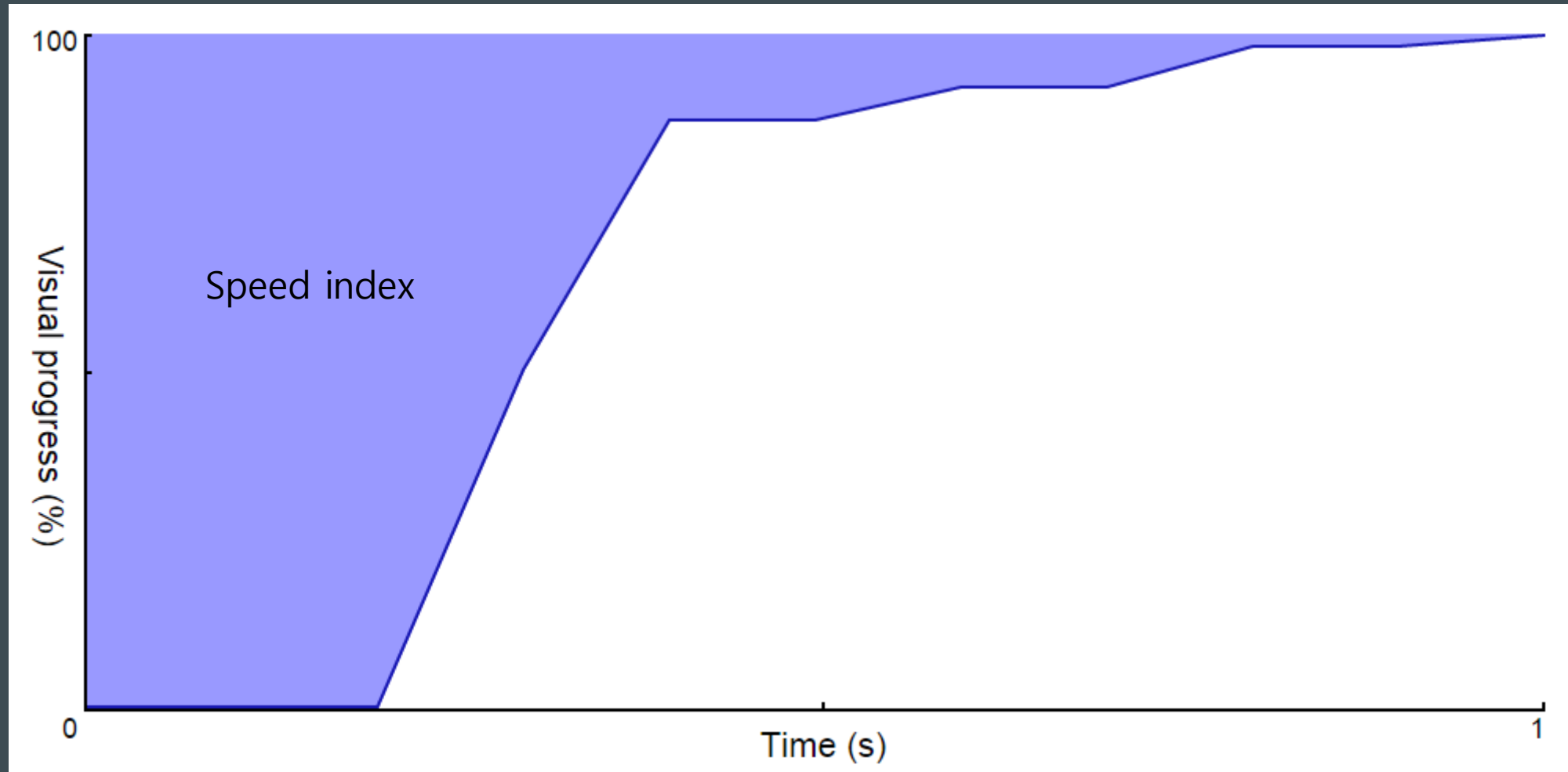


Speed Index 측정

페이지 로드 될 때까지 시간 경과에 따라 시각적으로 완료되는 양



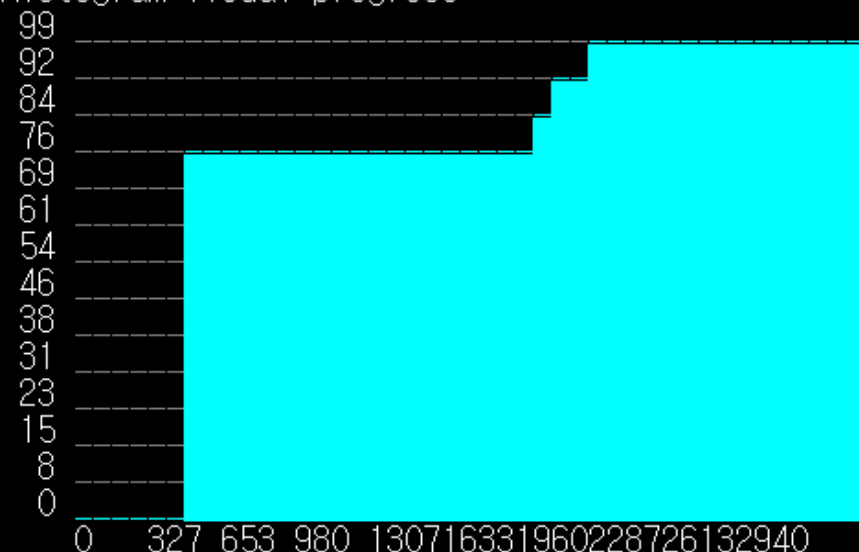
Speed Index 측정



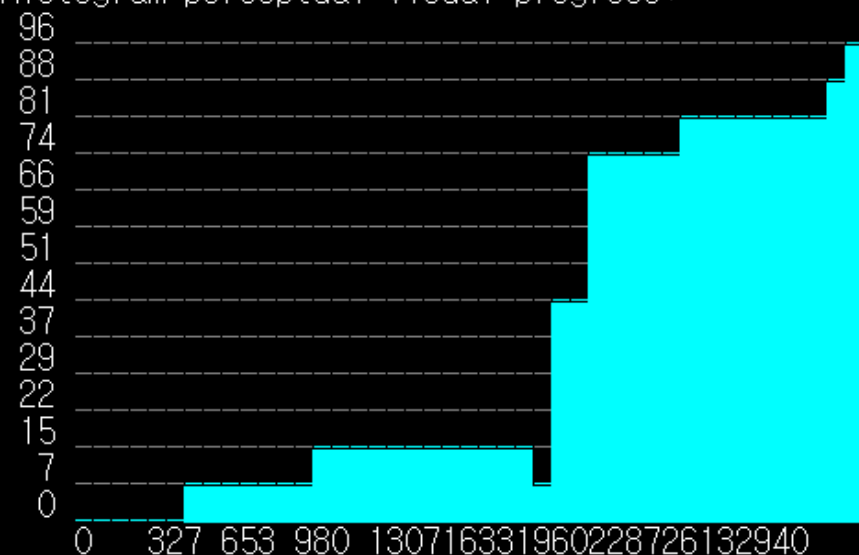
```
C:\boost\performance-history>speedline main-performance --pretty
Recording duration: 4253 ms (43 frames found)
First visual change: 533 ms
Last visual change: 3414 ms
Speed Index: 990.1
Perceptual Speed Index: 2247.0
```

Before

Histogram visual progress:



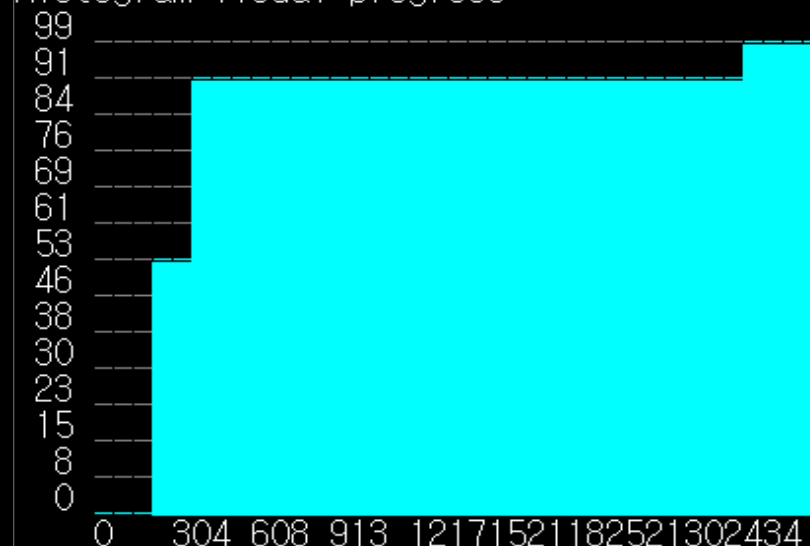
Histogram perceptual visual progress:



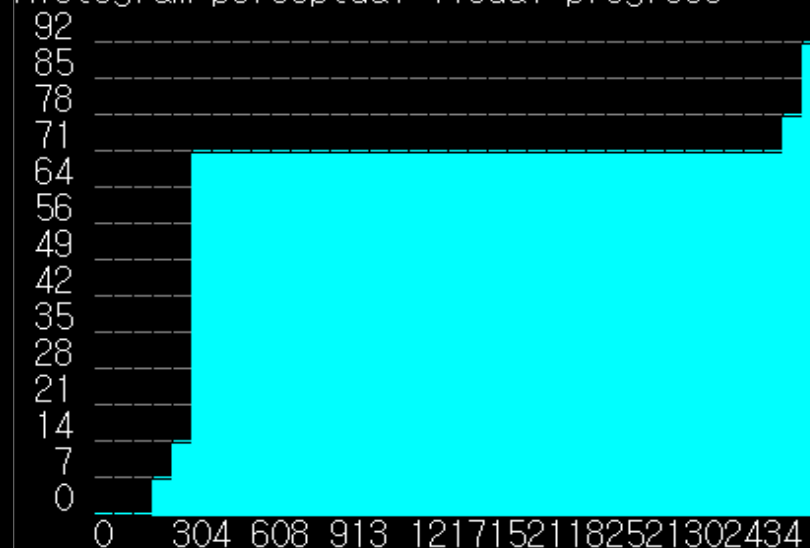
```
C:\Users\sollip\Desktop\Test>speedline local-si.json --pretty
Recording duration: 3281 ms (37 frames found)
First visual change: 240 ms
Last visual change: 2720 ms
Speed Index: 565.5
Perceptual Speed Index: 1128.3
```

After

Histogram visual progress:



Histogram perceptual visual progress:



Q&A

감사합니다.